The I in LLM stands for intelligence

January 2, 2024 Daniel Stenberg

I have held back on writing anything about AI or how we (not) use AI for development in the curl factory. Now I can’t hold back anymore. Let me show you the most significant effect of AI on curl as of today – with examples.

Bug Bounty

Having a bug bounty means that we offer real money in rewards to hackers who report security problems. The chance of money attracts a certain amount of “luck seekers”. People who basically just grep for patterns in the source code or maybe at best run some basic security scanners, and then report their findings without any further analysis in the hope that they can get a few bucks in reward money.

We have run the bounty for a few years by now, and the rate of rubbish reports has never been a big problem. Also, the rubbish reports have typically also been very easy and quick to detect and discard. They have rarely caused any real problems or wasted our time much. A little like the most stupid spam emails.

Our bug bounty has resulted in over 70,000 USD paid in rewards so far. We have received 415 vulnerability reports. Out of those, 64 were ultimately confirmed security problems. 77 of the report were *informative*, meaning they typically were bugs or similar. Making 66% of the reports neither a security issue nor a normal bug.

Better crap is worse

When reports are made to *look* better and to *appear* to have a point, it takes a longer time for us to research and eventually discard it. Every security report has to have a human spend time to look at it and assess what it means.

The better the crap, the longer time and the more energy we have to spend on the report until we close it. A crap report does not help the project at all. It instead takes away developer time and energy from something productive. Partly because security work is consider one of the most important areas so it tends to trump almost everything else.

A security report can take away a developer from fixing a really annoying bug. because a security issue is always more important than other bugs. If the report turned out to be crap, we did not improve security and we missed out time on fixing bugs or developing a new feature. Not to mention how it drains you on energy having to deal with rubbish.

AI generated security reports

I realize AI *can* do a lot of good things. As any general purpose tool it can also be used for the wrong things. I am also sure AIs *can* be trained and ultimately get used even for finding and reporting security problems in productive ways, but so far we have yet to find good examples of this.

Right now, users seem keen at using the current set of LLMs, throwing some curl code at them and then passing on the output as a security vulnerability report. What makes it a little harder to detect is of course that users copy and paste and include their own language as well. The entire thing is not exactly what the AI said, but the report is nonetheless crap.

Detecting AI crap

Reporters are often not totally fluent in English and sometimes their exact intentions are hard to understand at once and it might take a few back and fourths until things reveal themselves correctly – and that is of course totally fine and acceptable. Language and cultural barriers are real things.

Sometimes reporters use AIs or other tools to help them phrase themselves or translate what they want to say. As an aid to communicate better in a foreign language. I can’t find anything wrong with that. Even reporters who don’t master English can find and report security problems.

So: just the mere existence of a few give-away signs that parts of the text were generated by an AI or a similar tool is not an immediate red flag. It can still contain truths and be a valid issue. This is part of the reason why a well-formed crap report is harder and takes longer to discard.

Exhibit A: code changes are disclosed

In the fall of 2023, I alerted the community about a pending disclosure of CVE-2023-38545. A vulnerability we graded severity high.

The day before that issue was about to be published, a user submitted this report on Hackerone: Curl CVE-2023-38545 vulnerability code changes are disclosed on the internet

That sounds pretty bad and would have been a problem if it actually was true.

The report however reeks of typical AI style hallucinations: it mixes and matches facts and details from old security issues, creating and making up something new that has no connection with reality. **The changes had not been disclosed on the Internet**. The changes that actually had been disclosed were for previous, older, issues. Like intended.

In this particular report, the user helpfully told us that they used Bard to find this issue. Bard being a Google generative AI thing. It made it easier for us to realize the craziness, close the report and move on. As can be seen in the report log, we did have to not spend much time on researching this.

Exhibit B: Buffer Overflow Vulnerability

A more complicated issue, less obvious, done better but still suffering from hallucinations. Showing how the problem grows worse when the tool is better used and better integrated into the communication.

On the morning of December 28 2023, a user filed this report on Hackerone: Buffer Overflow Vulnerability in WebSocket Handling. It was morning in my time zone anyway.

Again this sounds pretty bad just based on the title. Since our WebSocket code is still experimental, and thus not covered by our bug bounty it helped me to still have a relaxed attitude when I started looking at this report. It was filed by a user I never saw before, but their "reputation" on Hackerone was decent – this was not their first security report.

The report was pretty neatly filed. It included details and was written in proper English. It also contained a proposed fix. It did not stand out as wrong or bad to me. It appeared as if this user had detected something bad and as if the user understood the issue enough to also come up with a solution. As far as security reports go, this looked better than the average first post.

In the report you can see my first template response informing the user their report had been received and that we will investigate the case. When that was posted, I did not yet know how complicated or easy the issue would be.

Nineteen minutes later I had looked at the code, not found any issue, read the code again and then again a third time. *Where on earth is the buffer overflow the reporter says exists here?* Then I posted the first question asking for clarification on where and how exactly this overflow would happen.

After repeated questions and numerous hallucinations I realized this was not a genuine problem and on the afternoon that same day I closed the issue as not applicable. **There was no buffer overflow.**

I don't know for sure that this set of replies from the user was generated by an LLM but it has several signs of it.

Ban these reporters

On Hackerone there is no explicit "ban the reporter from further communication with our project" functionality. I would have used it if it existed. Researchers get their "reputation" lowered then we close an issue as not applicable, but that is a very small nudge when only done once in a single project.

I have requested better support for this from Hackerone. **Update:** this function *exists*, I just did not look at the right place for it…

Future

As these kinds of reports will become more common over time, I suspect we might learn how to trigger on *generated-by-AI* signals better and dismiss reports based on those. That will of course be unfortunate when the AI is used for appropriate tasks, such as translation or just language formulation help.

I am convinced there will pop up tools using AI for this purpose that actually work (better) in the future, at least part of the time, so I cannot and will not say that AI for finding security problems is necessarily always a bad idea.

I do however suspect that if you just add an ever so tiny (intelligent) human check to the mix, the use and outcome of any such tools will become so much better. I suspect that will be true for a long time into the future as well.

I have no doubts that people will keep trying to find shortcuts even in the future. I am sure they will keep trying to earn that quick reward money. Like for the email spammers, the cost of this ends up in the receiving end. The ease of use and wide access to powerful LLMs is just too tempting. I strongly suspect we will get more LLM generated rubbish in our Hackerone inboxes going forward.

Discussion

Hacker news

Credits

Image by Haider Mahmood from Pixabay